**Lab 03 - Accounts, Permissions, and Network Tools**

# Introduction

In this lab, we will take a look at user accounts and passwords, file permissions, and several applications that will allow us to analyze our network. These tools will become increasingly important as we progress through the semester, so take note to learn and understand them this week. For this lab, we will utilize the **desktop** machine that we built in the first lab, so it will not be necessary to create a new machine this week. [Don’t forget to follow these lab report instructions.](https://docs.google.com/document/d/17YVskN9GQxI6Erjl8pq7goWKIcbaZ7PeJeYBoVwNsKg/edit?usp=sharing)

# Users/Passwords

User accounts are assigned to each user/process needing to interact with the system. The main components that create a user account are: username, group, UID, GID, password, home directory, and shell. The username and group are used by humans to distinguish accounts, while the UID and GID are used by the system. Each account also belongs to one or more groups, which can be used to grant certain privileges to various groups of users. The password is used to authenticate users, the home directory storage set aside for the user to use, and the shell dictates which program to run once a user has been authenticated (usually a shell of some sort).

Each user account is maintained in a set of files located in /etc/: passwd, shadow, group, gshadow.

* passwd - lists each account associated with the system, the UID and GID (unique identifiers), full name, the location of their home directory, and the default shell to be used when logging in (/usr/sbin/nologin dictates accounts that should have no login permissions).
* group - lists each group associated with the system and the corresponding GID.
* shadow - lists each user account and the corresponding password hash. Additional information such as last password change, date of expiration, etc. is included.
* gshadow - lists each group and corresponding group password hash (if it exists).

These files combined with file permissions dictate what a user can and can’t do. Users can be created by an administrator with the adduser command.

Using online resources and/or lecture notes, take a screenshot of your passwd and shadow files and notate which “fields” correspond to which bit of information for your own user account (should be near the bottom of file). You will need both the screenshots and the annotations to receive credit. Include this in your lab report.

### Creating a User

Create a new user:

sudo adduser test\_user

Look at the console messages and notice how it creates a user, creates a group, adds the user to the group, and creates the user's home directory. The numbers listed in parentheses after the username and group name are the UID and GID.

Give the new user a password. You will not be able to see the password as you type. We recommend setting the password to *password* (easy to remember this way). Leave all other information blank and finish the wizard.

The adduser command updates the aforementioned files to contain the specified username and new group for the user (same group/GID as username/UID).

Take a screenshot of /etc/passwd and /etc/shadow after creating the user, but **before following the next steps.** Include this in your lab report.

To see that the new account has been created, switch to the new user by using su(substitute user).

su - test\_user

Now that you've logged in, navigate to your home directory (/home/test\_user or ~ for short) and run the following command.

ls -la

This lists all (-a) your files in the "long" (-l) listing format with more details. You should see that adduser has provisioned test\_user's new home folder with some default hidden configuration files.

Run the following command to show the USER and SHELL environment variables that were loaded from passwd. This shows that your previous su command succeeded in logging you in.

echo $USER $SHELL

Take a screenshot of the detailed directory listing of test\_user's home directory and the output of the environment variables. Include this in your lab report.

Type exit to leave test\_user's shell session and return to your own.

### Changing a User's Login Shell

The chsh (**ch**ange login **sh**ell) command can change a user's shell. Run the following command to set test\_user's shell to nologin, which is a "shell" that rejects the user's attempt to log in.

sudo chsh -s /usr/sbin/nologin test\_user

Now, use su - again to try to log in as test\_user. You should see that test\_user can no longer log in to an interactive shell session.

### Changing Passwords

The passwd command allows you to change user account passwords. In addition to changing passwords, it can delete a user's password (passwordless account), lock a user's password, and more. Using the man pages for nologin and passwd (man nologin and man passwd), research the difference between setting a user's shell to nologin and locking a user's password (passwd -l).

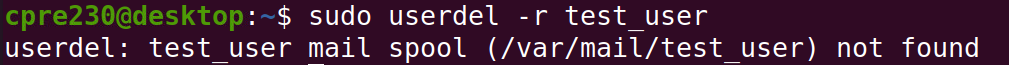
In your lab report, summarize the key differences you found between nologin and a locked password.

### Deleting a User Account

After completing the exercises above, remove the account with the following

sudo userdel -r test\_user

You will see the message below. It is not an error to worry about. It just means that there wasn't a mail folder created for the user. Therefore, when “delete user” ran, it reported that it couldn't find that file.



# Permissions

In a Linux environment, permissions play a crucial role in determining what a user is able or unable to access, modify, or create. Each file has a set of metadata associated with it in order to determine who can do what to it. We will use the command ls -lan test.txt where ls stands for list directory contents and the flags are -l for using the long listing format, -a for all files (do not ignore the ones starting with .) and -n for listing the numeric user and group IDs.

As a quick review, note that permissions take the following format.



Here we can see 10 characters, which dictate the object type (1 character - file, directory, etc.) and permissions (the last 9 characters), the number of hard links to this object (1), the owner (1000), the group (1000), the size in bytes, the time and date of last modification, and the object name. Although lots of information is provided here, for right now we are really only interested in the permissions, the owner, and the group. Notice that the UID and GID are displayed - this is to show that deep down, the system uses UIDs and GIDs to reference users (so as to mitigate headaches when renaming a user).

The permissions can be broken down into three sections, each with three characters. The first three describe the *user*, the second three the *group*, and the last three the *world*. The characters in each section tell, in order, whether the corresponding entity can: read (r), write (w) and/or execute (x).

For example, in the screenshot above, we notice that the owner can read the file and write to the file (make changes), however cannot execute the file. The group and world can only read the file - meaning it cannot be written to or executed. Or, we could say that only the *user* can write to the file, the *assigned group* and *anybody* can read the file, and *nobody* can execute the file.

### Setting Ownership

To change the assigned user and group of a file or directory, we can use the chown (change ownership) command. The previous user's and group's permissions will be transferred to the new user and group, and the previous user and group will inherit the permissions applied to world. chown is typically used with the desired ownership in the format of user:group. chown generally needs to be run as root or with sudo.

Here are a few examples of changing the owner and group of test.txt:

sudo chown root:root test.txt

sudo chown sstudent:sstudent test.txt

sudo chown sstudent:web-devs test.txt

### Setting Permissions

To change the permissions of a file or directory, we can use the command chmod. This command is very flexible and can be used in many ways to achieve the same outcome. It is recommended that you read the manual page for chmod and use it to practice, ensuring that you understand its use.

Note that when using chmod**,** permissions assigned to an entity (user, group, world) can be thought of as a binary set of three digits, such that r=4, w=2, and x=1. If you want to give rwx then you add all the numbers together and that is the number you will use. You can use the command ls -l to view the permissions.

Here are some examples of use:

chmod 777 *filename*

chmod 700 *filename*

chmod u=rw *filename*

chmod go+x *filename*

chmod a+w *filename*

Use each of these commands on a file that you have in your home directory **(it is recommended that you make 5 copies of a file so that you have consistent starting permissions)** and include in the lab report how each of these commands modifies the file permissions. Make sure to be specific about which entity’s permission changed and which were not changed when using each command.

### Exploring Permissions

Now create a new directory and view what its permissions are by default. Using chmod, ls -l, and touch (to create files within), and any other commands you can think of, try to figure out what various permissions do for a **directory and files within it**. Include your answer in your lab report in an organized and understandable manner.

I recommend setting one option at a time and seeing what you can do to files in that directory. Some helpful things to try: Can you read the files? Can you copy the files? Can you list the contents of the directory? Can you create/delete files in that directory? Can you cd into the directory? A grid might be useful for keeping track of what can and can’t be done.

List the permissions for the **/etc/shadow** file and include who is able to read, write, and execute this file. Also include why you think the permissions are set this way.

# Network Tools

### ping

1. ping is an extremely useful tool for sending ICMP echo-request packets to network hosts (other servers, routers, etc.). We’ve already used this tool in the first couple of labs to make sure we had an established connection to our gateway and proxy server.
2. ping will automatically send an infinite number of packets to the host you specify, unless of course you use ctrl+c to stop ping early. Use the man pages to look up the ping command, and figure out what flag you need to add to send a specific number of packets (aka, ping will automatically stop after sending those packets). *Note what flag you need, as you will need this information for later in the lab.*

### dig

1. dig is a great tool for making queries to internet name servers; i.e. mapping domain names to IP addresses and vice versa.
2. Go ahead and type dig iastate.edu into the command line of your Ubuntu **desktop machine**. You’ll notice that a lot of information is returned. In the ANSWER section, you should be able to see the IP address of iastate.edu.
   1. Sometimes, the amount of information returned via the dig command can be very overwhelming and confusing. Luckily, dig allows us to "filter" out the sections that we don’t want. For example, to view only the answer section, type the following command:  
       dig iastate.edu +noall +answer
   2. This tells dig to say "no" to showing all of the sections, and then explicitly show only the answer section, which gives us the IP address of iastate.edu, what we are ultimately after.
3. dig also works the other way around, in what we call a “reverse lookup”. Try:

dig -x <IP returned in step b>

1. You should see that the reverse DNS needn't be the same as the forward DNS entry.
2. If you attempt to do a reverse lookup on your own IP address (dig -x <your IP address> +noall +answer), you’ll notice that nothing is returned, as the internet doesn’t know what to call your server. We will build a DNS server in lab next week, which will bring that functionality to our network!
3. dig advanced queries
   1. With dig, there are all sorts of specific queries that can be made to request certain information. Please take the time to search the internet for what the following queries mean, or you can try them out yourself on this **desktop machine**. Please include this table with your pdf submission.

| Query | What does the query result mean? |
| --- | --- |
| dig -t mx iastate.edu |  |
| dig -t ns iastate.edu |  |
| dig -t soa iastate.edu |  |
| dig -t aaaa iastate.edu |  |
| dig -t any iastate.edu |  |

### whois

1. whois can be used to return lots of information concerning a specific domain. Information such as registrant, administrator, and technical contact information can be found (and potentially more, depending on the domain).
2. This information must be provided to the domain registrar when purchasing a domain and might reveal quite a lot of information about the business/person who purchases the domain.
3. Services called “domain privacy” can be obtained and used to provide vague information for the publically facing whois lookup information (the company offering the privacy uses its own information in place of the owner’s). The domain registrar still contains the real information, but the owner is allowed to obscure much of the publicly facing information.
4. Try running whois from [ICANN whois](https://whois.icann.org/en/lookup?name=) on a few different websites and see what kind of information you can find - try finding a few that have protection and a few that don’t.

### traceroute

1. traceroute is a tool that is used for tracing the path a network packet takes, from source to destination. It is not installed by default on your desktop. So, first you need to install the package

sudo apt install traceroute

1. Because ISEAGE contains all traffic inside it, using traceroute on an external IP can lead to some interesting results. Instead, try tracing the path to the proxy server:

traceroute 199.100.16.100

1. Note how the packet travels through your default gateway to reach the proxy.

### netstat

1. netstat is a great tool for showing the open sockets and listening ports on your machine. It shows details about open ports, listening addresses, associated processes, etc. However, it also needs to be installed (fyi...by default another command called ss is installed and it also works like netstat).

sudo apt install net-tools

1. Unfortunately, since our desktop and server machines aren’t really serving any purpose/offering any services, performing netstat currently won’t yield anything useful. So, we will use this awesome tool called ***netcat*** to temporarily open ports on our machines. **On your ns1 server**, type the following command:

sudo netcat -l 22

1. This will open up port 22. Press ctrl+z to switch netcat to a background process, a term you should remember from a previous lab.
2. Now that we’ve opened this port, go ahead and perform the following two netstat commands. For this class, we will be using the following flags (**Still on your ns1 server**):

netstat -tl

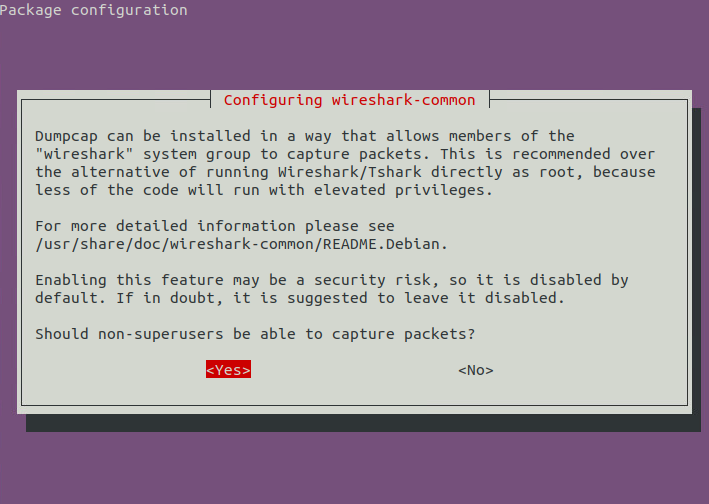
netstat -tln

1. Take a screenshot of the output of both of these commands, and include this screenshot with your pdf submission
2. Also for this lab submission, determine what each of the flags is telling netstat to do (using the man pages). Include this in your lab pdf.

| Flag | Meaning |
| --- | --- |
| -t |  |
| -l |  |
| -n |  |

# Installing and Using Wireshark

* 1. Wireshark is a handy GUI tool that you can use to monitor network traffic in real time. For the purposes of this lab, we will be using Wireshark to capture HTTP web traffic.
  2. Install Wireshark **(on your desktop machine)**  
      sudo apt install wireshark
  3. Select "yes" to the question about allowing access to non-superusers.



* 1. Next, enter the following command, to allow your user permission to utilize Wireshark (default is that only root/sudo users can monitor traffic):

sudo usermod -a -G wireshark $USER

* 1. After entering this command, **you MUST log out and then log back in again** (or reboot) to allow Wireshark to capture packets
  2. Search for Wireshark in the Ubuntu "Activities" search bar and open the application.
  3. Start Wireshark by double-clicking on ens160 to set it listening on this interface
  4. Type http into "Apply a display filter…" and press enter to filter traffic to *only* http traffic.
  5. Next, open up Firefox and navigate to http://www.iastate.edu
  6. Go back into Wireshark, and you should see the captured http traffic. Take a screenshot of your findings to include in your lab submission.
  7. When you are done capturing with Wireshark, don't forget to close Wireshark or stop capturing packets with the red stop button.

# tcpdump - The Terminal Equivalent to Wireshark

1. For the last portion of this lab, we will use the tool tcpdump to observe filtered traffic through the command line. Instead of watching for http traffic, however, we will filter by the ICMP protocol.
2. Open up a terminal on your Ubuntu Desktop machine. Type in the following command:

sudo tcpdump -i ens160 icmp

1. This command runs the tcpdump process over the ens160 interface (our default network interface) and filters traffic by ICMP packets only.
2. Now, *while leaving the current terminal window running*, open up a new terminal window, so that they are sitting side-by-side. Go ahead and use ping to send 5 ICMP packets to your Ubuntu server (should be XXX.XXX.XXX.200)
   1. \*See the ping section above if you are unsure how to do this, or look up ping in the man pages (man ping)\*
3. You should see the results of your ping in your tcpdump process running in the other terminal window. Please take a screenshot of this, and add it to your pdf submission

# 

# 

# 

# 

# 

# 

# 

# 

# 

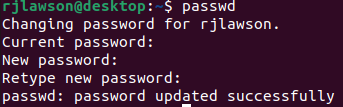
# 

# Lab 03 Template

[Don’t forget to follow these lab report instructions.](https://docs.google.com/document/d/17YVskN9GQxI6Erjl8pq7goWKIcbaZ7PeJeYBoVwNsKg/edit?usp=sharing)

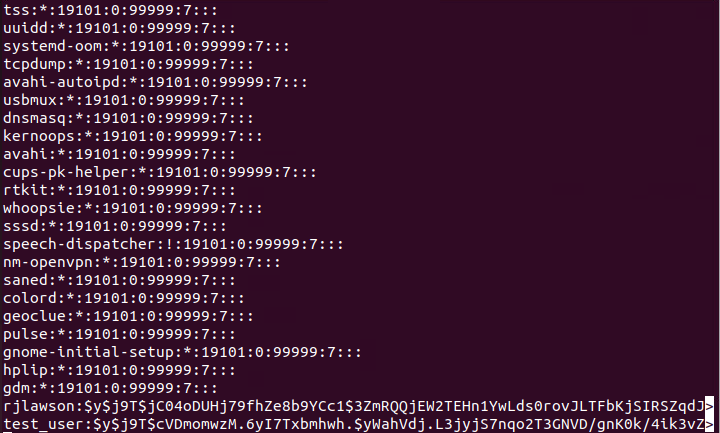
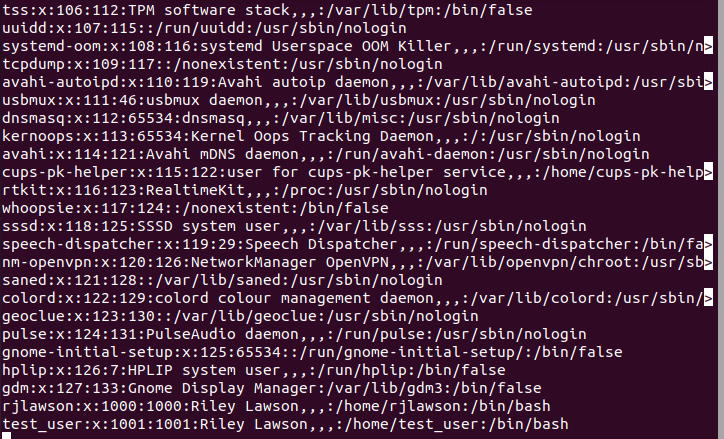
1. **Screenshot of passwd and shadow with description/notation of fields**

(10 points)



1. **Screenshot of passwd and shadow in middle of user account creation**

(10 points)

****

1. **Screenshot of new user’s home directory contents (ls -la) and the output of the environment variables (echo $USER $SHELL)**

(5 points)

1. **Summarize the Key Differences between nologin and a locked password**

(5 points)

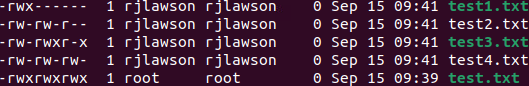
The difference between nologin and a locked password is that when nologin is enabled it is impossible to login to that account (not available) {Politely refuse a login} whereas a locked password can be cracked if it is hacked or a user has a proper authentication key. This also means that the user disables the password and changes it to an encrypted value.

1. **Description of the effect of each of the (5) chmod commands**

(10 points)

chmod modifies the permission of files/directories.

1. 777 gives full read, write, and execute permissions to the file for the user and root
2. 700 gives read, write, execute permissions to only the root user
3. u=rw gives read, write permissions to the person who own the file
4. go+x gives execute permissions to the group and other users
5. a+w sets write permissions to 7 for example. This modifies the bits to 111



1. **Organized description of directory permissions and how affects contents**

(10 points)



r = read

w = write

x = execute

This affects content because it can allow you to read only, write only, write and execute files, or execute only.

1. **Screenshot and description of /etc/shadow file - who can r/w/x and why it’s set this way.**

(10 points)

****

Nobody can execute this file, because there is nothing to execute, you can read and write it, but only as root. This is because through this file you could view an encrypted password which would allow you to decrypt and change the current password.

1. **Meanings of dig queries**

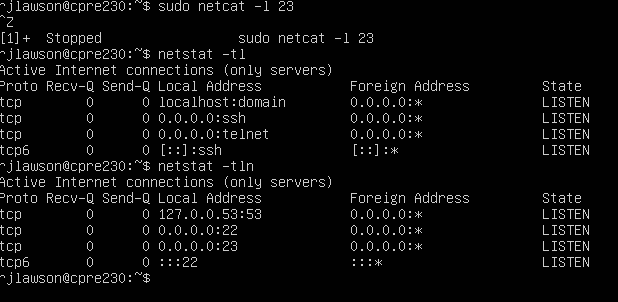
(10 points)

The meaning of dig queries are supposed to be a tool to allow you that gives you related information about whatever DNS you're looking up.

| Query | What does the query result mean? |
| --- | --- |
| dig -t mx iastate.edu | MX means that it’ll specify a mail server |
| dig -t ns iastate.edu | NS means that it lists DNS servers |
| dig -t soa iastate.edu | SOA means start of authority and includes DNS, server name, email of domain admin, serial number, and timers |
| dig -t aaaa iastate.edu | Doesn’t display anything, however this should display a ipv6 address |
| dig -t any iastate.edu | Displays all record types |

1. **Screenshot of the (2) netstat outputs**

(10 points)

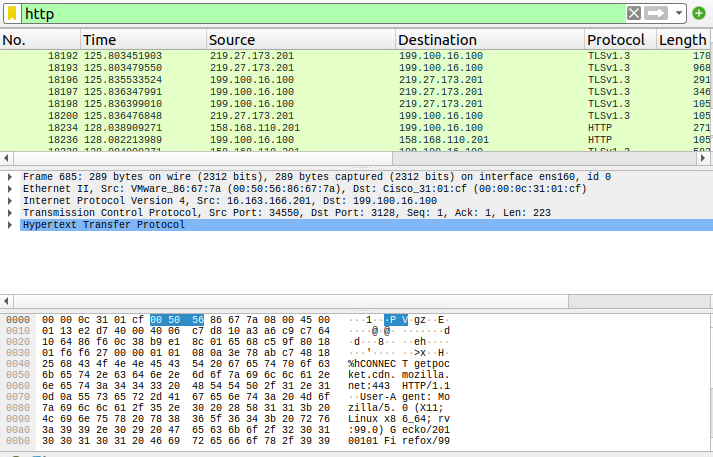
****

1. **Complete the netstat flag table**

(10 points for all 3 correct meanings; 0 points for anything less)

| Flag | Meaning |
| --- | --- |
| -t | -t = tcp |
| -l | -l = listening, only listening sockets |
| -n | -n = numeric, shows numerical addresses |

1. **Screenshot of the captured http traffic using Wireshark**

(5 points)

1. **Screenshot of the captured icmp traffic using tcpdump**

(5 points)